IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Applicants:	Rikin S. Patel	§	Art Unit:	2192
		§		
Serial No.:	10/694,518	§	Confirmation No.:	1895
		§		
Filed:	10/27/2003	§	Examiner:	Thuy Chan Dao
		§		
For:	TRANSACTION PROCESSING ARCHITECTURE	§	Atty. Dkt. No.:	200901488-2 (HPC.0845US)
		§		
		§		
		§		

Mail Stop Appeal Brief-Patents
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313-1450

APPEAL BRIEF PURSUANT TO 37 C.F.R § 41.37

Sir:

The final rejection of claims 1-30 is hereby appealed.

I. REAL PARTY IN INTEREST

The real party in interest is the Hewlett-Packard Development Company, LP. The Hewlett-Packard Development Company, LP, a limited partnership established under the laws of the State of Texas and having a principal place of business at 11445 Compaq Center Drive West, Houston, TX 77707, U.S.A. (hereinafter "HPDC"). HPDC is a Texas limited partnership and is a wholly-owned affiliate of Hewlett-Packard Company, a Delaware Corporation, headquartered in Palo Alto, CA. The general or managing partner of HPDC is HPQ Holdings, LLC.

II. RELATED APPEALS AND INTERFERENCES

None.

III. STATUS OF THE CLAIMS

Claims 1-30 have been finally rejected and are the subject of this appeal.

IV. STATUS OF AMENDMENTS

No amendment after the final rejection of March 26, 2009 has been submitted. Therefore, all amendments have been entered.

V. SUMMARY OF THE CLAIMED SUBJECT MATTER

The following provides a concise explanation of the subject matter defined in each of the independent claims involved in the appeal, referring to the specification by page and line number and to the drawings by reference characters, as required by 37 C.F.R. § 41.37(c)(1)(v). Each element of the claims is identified by a corresponding reference to the specification and drawings where applicable. Note that the citation to passages in the specification and drawings for each claim element does not imply that the limitations from the specification and drawings should be read into the corresponding claim element.

Independent claim 1 recites a schema generator, comprising:

a computer readable storage medium (Fig. 1:14, 16, 22; Spec., p. 5, ln. 12-16);

computer software stored on the computer readable storage medium and operable to (Spec., p. 5, ln. 17-19; p. 6, ln. 1-9):

parse a plurality of transaction definitions (Fig. 2:32) for a software system, wherein each transaction definition comprises one or more parameters (Spec., p. 9, ln. 3-5); and

generate, in response to parsing the plurality of transaction definitions, a plurality of schema definitions (Fig. 2:36) for at least a portion of the parsed transaction definitions, wherein the schema definitions are written in a self-describing language (Spec., p. 7, ln. 16 – p. 8, ln. 21; p. 9, ln. 5 – p. 10, ln. 2);

wherein a first schema definition is operable to map the one or more parameters associated with a first transaction definition to a first document written in the self-describing language (Spec., p. 9, ln. 15-20); and

wherein a second schema definition is operable to map a second document written in the self-describing language to the one or more parameters associated with a second transaction definition (Spec., p. 9, ln. 15-20).

Independent claim 8 recites a method for generating a plurality of schema definitions, comprising:

parsing a plurality of transaction definitions (Fig. 2:32) for a software system, wherein each transaction definition comprises one or more parameters (Spec., p. 9, ln. 3-5); and

generating, in response to parsing the plurality of transaction definitions, a plurality of schema definitions (Fig. 2:36) for at least of portion of the parsed transaction definitions, wherein the schema definitions are written in a self-describing language (Spec., p. 7, ln. 16 – p. 8, ln. 21; p. 9, ln. 5 – p. 10, ln. 2);

wherein a first schema definition is operable to map the one or more parameters associated with a first transaction definition to a first document written in the self-describing language (Spec., p. 9, ln. 15-20); and

wherein a second schema definition is operable to map a second document written in the self-describing language to the one or more parameters associated with a second transaction definition (Spec., p. 9, ln. 15-20).

Independent claim 12 recites a transaction processing system comprising:

one or more processing units (Spec., 11:29 – p. 12, ln. 6; Fig. 1: 10, 12; Spec., p. 5, ln. 12-13) operable to execute:

a software service (Fig. 3:60) operable to receive a transaction request and to generate a first object (Fig. 3:62) associated with the transaction request (Spec., p. 11, ln. 9-27);

an object generator (Fig. 3:64) operable to convert the first object into a first document (Fig. 3:70) written in a self-describing language (Spec., p. 12, ln. 7-16); and

a document generator (Fig. 3:72) operable to convert the first document into a first transaction message (Fig. 3:76) according to a schema associated with a first transaction type determinable from the first document (Spec., p. 12, ln. 19-30).

Independent claim 22 recites a method for processing a transaction, comprising:

receiving a transaction request (Spec., p. 11, ln. 9-27);

generating a first object (Fig. 3:62) associated with the transaction request (Spec., p. 11, ln. 9-27);

converting the first object into a first document (Fig. 3:70) written in a self-describing language (Spec., p. 12, ln. 7-16); and

converting the first document into a first transaction message (Fig. 3:76) according to a schema associated with a first transaction type determinable from the first document (Spec., p. 12, ln. 19-30).

VI. GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

A. Claims 12-21 were rejected under 35 U.S.C. § 101.

B. Claims 1-30 were rejected under 35 U.S.C. § 102(e) as anticipated by Ankireddipally (U.S. Patent No. 6,772,216).

VII. ARGUMENT

The claims do not stand or fall together. Instead, Appellant presents separate arguments for various independent and dependent claims. Each of these arguments is separately argued below and presented with separate headings and sub-headings as required by 37 C.F.R. § 41.37(c)(1)(vii).

A. Claims 12-21 were rejected under 35 U.S.C. § 101.

1. Claims 12-21.

Independent claim 12 was rejected under 35 U.S.C. § 101 as purportedly being directed to non-statutory subject matter. This rejection was based on the Examiner's incorrect assertion that "one or more processing units operable to execute" can read merely on software. 3/26/2009 Office Action at 4. As purported support for this reading of "one or more processing units operable to execute," the Examiner cited the schema generator 34 in Fig. 2 of the specification, as well as page 6, lines 17-29, and page 7, lines 16-33, of the specification. The cited page 6 passage refers to an architecture that has components that may include software applications and data stored on one or more computers. Page 7, lines 16-33, of the specification provides a discussion of schema generator 34 that creates one or more schema 36. However, nowhere in these cited passages of the specification is there any definition of "processing units" as being merely software.

In fact, support for the subject matter of claim 12 is found at least in Fig. 3 and the accompanying text of the present specification. In the text accompanying Fig. 3, reference is made to various software residing on one or more **computers**. *See* Specification, page 11, line 29 – page 12, line 6. In addition, Fig. 1 of the specification shows an example general purpose computer 10 that has a processor 12. Thus, it is clear that "processing unit" as used in the claim refers to at least one computer or processor.

Note that claim 12 explicitly states that the one or more processing units are operable to **execute** various software components, including a software service, an object generator, and a document generator. A person of ordinary skill in the art would understand that such processing units would be hardware processing units, since a person of ordinary skill in the art would not typically refer to one software unit executing another software unit.

In view of the foregoing, it is clear that claim 12 recites statutory subject matter, and does not merely recite descriptive material *per se* or computer programs representing computer listings *per se*, as alleged by the Examiner.

In view of the foregoing, it is respectfully submitted that claims 12-21 satisfy 35 U.S.C. § 101.

Reversal of the § 101 rejection of the above claims is respectfully requested.

B. Claims 1-30 were rejected under 35 U.S.C. § 102(e) as anticipated by Ankireddipally (U.S. Patent No. 6,772,216).

1. Claims 1-11.

It is respectfully submitted that claim 1 is clearly not anticipated by Ankireddipally. It is apparent that in making the rejection, the Examiner has applied a rather unreasonable interpretation of the claim language in mapping claim elements to elements of Ankireddipally.

Claim 1 recites, *inter alia*, the following elements:

- parse a plurality of transaction definitions for a software system, wherein each transaction definition comprises one or more parameters; and
- generate, in response to parsing the plurality of transaction definitions, a plurality of schema definitions for at least a portion of the parsed transaction definitions, wherein the schema definitions are written in a self-describing language;

The Examiner equated various messages sent by a requesting application of Ankireddipally (including the messages listed in table 3 in columns 19 and 20 and depicted in Fig. 13 of Ankireddipally) as constituting the transaction definitions recited in claim 1. A message such as request, reply, cancel, parse, and notify, that is sent from a requesting application is merely a request to perform some action. It is unreasonable to construe a requesting message as being a transaction definition. In fact, Ankireddipally actually uses the term "transaction definition" in column 12. *See* Ankireddipally, 12:4-21. Thus, Ankireddipally provides objective evidence that a request message such as any of the messages depicted in table 3 or Fig. 13 of Ankireddipally is **not** the same as a transaction definition. It is clear that the transaction definitions that are discussed in Ankireddipally are not used to generate a schema definition as recited in claim 1.

Moreover, even if, for the sake of argument, the various request messages can be considered transaction definitions, it is respectfully submitted that these request messages are not parsed to generate schema definitions that are written in a self-describing language. The Examiner equated the XML documents 40 discussed in Ankireddipally as being the "schema definitions" recited in claim 1. As explained by Ankireddipally, a CX server 10 is a document-centric process automation application that exchanges messages in the form of XML documents 40 between CX clients. *Id.*, 12:33-36. Column 15 of Ankireddipally provides further explanation regarding the use of the XML documents, which are employed to pass information

between CX clients in the CX server 10. An XML document as described in Ankireddipally is thus a container for carrying information between different entities—the XML document cannot be considered a schema definition, as recited in claim 1. In fact, Ankireddipally itself employs the term "schema" in other contexts, which provides evidence that a person of ordinary skill in the art would understand that the XML documents 40 of Ankireddipally for exchanging information among users are different from schema definitions. For example, column 13 of Ankireddipally states that persistence service 19 has the responsibility of mapping between an XML document and a respective data store schema. *Id.*, 13:1-3. Thus, Ankireddipally explicitly provides a distinction between a "schema" and the XML document that is used as a container for information exchanged between clients through the CX server.

In view of the foregoing, it is clear that Ankireddipally clearly does not anticipate the subject matter of claim 1 and its dependent claims.

Similarly, independent claim 8 and its dependent claims are not anticipated by Ankireddipally.

Reversal of the final rejection of the above claims is respectfully requested.

2. Claims 12-21.

Independent claim 12 recites a transaction processing system comprising:

- a software service operable to receive a transaction request and to generate a first object associated with the transaction request;
- an object generator operable to convert the first object into a first document written in a self-describing language; and
- a document generator operable to convert the first document into a first transaction message according to a schema associated with a first transaction type determinable from the first document.

It is respectfully submitted that claim 12 is also clearly not anticipated by Ankireddipally. The last two elements of claim 12 recites an object generator operable to convert the first object

into a first document written in a self-describing language, and a document generator operable to convert the first document into a first transaction message according to a schema associated with a first transaction type determinable from the first document.

With respect to the "object generator" element of claim 12, the Examiner cited blocks 540 and 550 in Fig. 17 of Ankireddipally. 3/26/2009 Office Action at 11. In addition, with respect to the "document generator" element of claim 12, the Examiner cited the **same** blocks 540 and 550 of Fig. 17 of Ankireddipally. Block 540 in Fig. 17 of Ankireddipally refers to formatting output data into a correct protocol message format, and block 550 of Ankireddipally refers to sending a message with output results data to a client application. The Examiner appears to have equated the formatting element (block 540) in Fig. 17 of Ankireddipally as constituting "convert[ing] the first object into a first document written in a self-describing language," as recited in the "object generator" element of claim 12 If that is the case, block 550 in Fig. 17 of Ankireddipally clearly does not constitute a document generator converting the first document into a first transaction message according to a schema associated with a first transaction type determinable from the first document. All block 550 of Fig. 17 performs is to send a message with output results data to a client application. There is no teaching that the formatted output results (produced by block 540) is **converted** into a transaction message **according to a schema associated with a first transaction type determinable from the first document**.

In fact, blocks 532, 540, and 550 shown in Fig. 17 of Ankireddipally refer to various messages exchanged in Fig. 16 of Ankireddipally. As explained in column 21 of Ankireddipally, after formatting the output data into the correct protocol message in block 540 of Fig. 17, a protocol plug-in component 224 (shown in Fig. 16 of Ankireddipally) sends a reply message 340

(in block 550) to the requesting application 214. It is clear that there is no conversion of the formatted output results (produced at block 540) in to a first transaction message according to a schema associated with a first transaction type determinable from the first document.

With respect to the "according to a schema" language of claim 12, the Examiner cited Fig. 13 and DTDs mentioned elsewhere in Ankireddipally. Fig. 13 shows processing of various request messages—there is absolutely no hint given in Fig. 13 of a document generator to convert the first document (in a self-describing language) into a first transaction message according to a schema associated with a first transaction type determinable from the first document. With respect to DTDs, the Examiner cited the following passage of Ankireddipally: column 16, line 21 – column 18, line 67. However, the teaching regarding DTDs in Ankireddipally does not remedy the deficiency that block 550 in Fig. 17 of Ankireddipally does not perform any conversion of a document written in a self-describing language into a transaction message according to a schema associated with a transaction type determinable from the first document.

In view of the foregoing, it is clear that claim 12 and its dependent claims are not anticipated by Ankireddipally.

Reversal of the final rejection of the above claims is respectfully requested.

3. Claims 22-30.

Independent claim 22 recites a method for processing a transaction that comprises:

- receiving a transaction request;
- generating a first object associated with the transaction request;
- converting the first object into a first document written in a self-describing language; and
- converting the first document into a first transaction message according to a schema associated with a first transaction type determinable from the first document.

Claim 22 and its dependent claims are allowable over Ankireddipally for similar reasons as claim 12.

Reversal of the final rejection of the above claims is respectfully requested.

CONCLUSION

In view of the foregoing, reversal of all final rejections and allowance of all pending claims is respectfully requested.

Respectfully submitted,

Date: August 25, 2009

/Dan C. Hu/

Dan C. Hu
Registration No. 40,025
TROP, PRUNER & HU, P.C.
1616 South Voss Road, Suite 750
Houston, TX 77057-2631
Telephone: (713) 468-8880
Facsimile: (713) 468-8883

VIII. APPENDIX OF APPEALED CLAIMS

The claims on appeal are:

1. A schema generator, comprising:

a computer readable storage medium;

computer software stored on the computer readable storage medium and operable to:

parse a plurality of transaction definitions for a software system, wherein each transaction definition comprises one or more parameters; and

generate, in response to parsing the plurality of transaction definitions, a plurality of schema definitions for at least a portion of the parsed transaction definitions, wherein the schema definitions are written in a self-describing language;

wherein a first schema definition is operable to map the one or more parameters associated with a first transaction definition to a first document written in the self-describing language; and

wherein a second schema definition is operable to map a second document written in the self-describing language to the one or more parameters associated with a second transaction definition.

2. The schema generator of Claim 1, wherein the self-describing language comprises Extensible Markup Language (XML) or any version thereof.

3. The schema generator of Claim 1, wherein the self-describing language comprises HyperText Markup Language (HTML) or any version thereof.

4. The schema generator of Claim 1, wherein the self-describing language comprises a language that employs hypertext.

5. The schema generator of Claim 1, wherein the software system comprises an Information Management System (IMS).

6. The schema generator of Claim 1, wherein the transaction definitions are associated with a message format service.

7. The schema generator of Claim 6, wherein the self-describing language comprises Extensible Markup Language (XML) or any version thereof.

8. A method for generating a plurality of schema definitions, comprising:

parsing a plurality of transaction definitions for a software system, wherein each transaction definition comprises one or more parameters; and

generating, in response to parsing the plurality of transaction definitions, a plurality of schema definitions for at least of portion of the parsed transaction definitions, wherein the schema definitions are written in a self-describing language;

wherein a first schema definition is operable to map the one or more parameters associated with a first transaction definition to a first document written in the self-describing language; and

wherein a second schema definition is operable to map a second document written in the self-describing language to the one or more parameters associated with a second transaction definition.

9. The method of Claim 8, wherein the self-describing language comprises Extensible Markup Language (XML) or any version thereof.

10. The method of Claim 8, wherein the self-describing language comprises HyperText Markup Language (HTML) or any version thereof.

11. The method of Claim 8, wherein the transaction definitions are associated with a message format service.

12. A transaction processing system comprising:

one or more processing units operable to execute:

a software service operable to receive a transaction request and to generate a first object associated with the transaction request;

an object generator operable to convert the first object into a first document written in a self-describing language; and

a document generator operable to convert the first document into a first transaction message according to a schema associated with a first transaction type determinable from the first document.

13. The transaction processing system of Claim 12, wherein the self-describing language comprises Extensible Markup Language (XML) or any version thereof.

14. The transaction processing system of Claim 12, wherein the self-describing language comprises HyperText Markup Language (HTML) or any version thereof.

15. The transaction processing system of Claim 12, wherein the transaction generator is further operable to send the first transaction message to a message format service.

16. The transaction processing system of Claim 12, wherein the document generator is further operable to receive a second transaction message and convert the second transaction message into a second document according to a schema associated with a second transaction type determinable from the second transaction message; and

wherein the second document is written in the self-describing language.

17. The transaction processing system of Claim 16, wherein the object generator is further operable to convert the second document into a second object.

18. The transaction processing system of Claim 17, wherein the software service is further operable to receive the second object in response to the transaction request.

19. The transaction processing system of Claim 18, wherein the self-describing language comprises Extensible Markup Language (XML).

20. The transaction processing system of Claim 16, wherein the software service is further operable to receive the second document in response to the transaction request.

21. The transaction processing system of Claim 12, wherein the software service comprises a web service and wherein the definition of the first object has been published in a registry.

22. A method for processing a transaction, comprising:
receiving a transaction request;
generating a first object associated with the transaction request;
converting the first object into a first document written in a self-describing language; and
converting the first document into a first transaction message according to a schema associated with a first transaction type determinable from the first document.

23. The method of Claim 22, wherein the self-describing language comprises Extensible Markup Language (XML) or any version thereof.

24. The method of Claim 22, wherein the self-describing language comprises HyperText Markup Language (HTML) or any version thereof.

25. The method of Claim 22, further comprising:
sending the first transaction message to a message format service.

26. The method of Claim 22, further comprising:

receiving a second transaction message;

converting the second transaction message into a second document according to a schema associated with a second transaction type determinable from the second transaction message; and

wherein the second document is written in the self-describing language.

27. The method of Claim 26, further comprising:

converting the second document into a second object.

28. The method of Claim 27, further comprising:

receiving the second object in response to the transaction request.

29. The method of Claim 28, further comprising:

wherein the self-describing language comprises Extensible Markup Language (XML).

30. The method of Claim 22, wherein the first object is generated by a web service and wherein the definition of the first object has been published in a registry.

IX. EVIDENCE APPENDIX

None.

X. RELATED PROCEEDINGS APPENDIX

None.